Attorney's Docket No.: 34874-020/2003P00061US01
Customer No.: 64280

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No.:		10/628,959	Conf. No.:	6174
Applicant	:	Eitan Hefetz	TC/Art Unit:	2178
Filed	:	July 28, 2003	Examiner:	Manglesh M. Patel
For	:	SELECTIVELY INTERPRETED PORTAL PAGE LAYOUT TEMPLATE		

Mail Stop: APPEAL BRIEF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

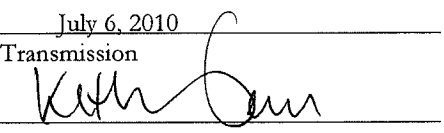
APPEAL BRIEF

Appellants file this Appeal Brief pursuant to 37 C.F.R. § 41.37 in support of an appeal of the final rejection of claims 1-20 under 35 U.S.C. §102(B) in the Final Office Action of January 5, 2010. The fee of $540.00 specified under 37 C.F.R. §41.20(b) for filing of a Brief in Support of an Appeal by other than a small entity is submitted herewith. Appellants are concurrently filing herewith a Petition for a one-month extension of time, authorization for a credit-card payment of the filing fees mentioned above is submitted herewith. No additional fees are believed to be due, however, the Commissioner is hereby authorized to charge any additional fees that may be due, or credit any overpayment of same, to Deposit Account No. 50-0311.

CERTIFICATE OF ELECTRONIC TRANSMISSION

I hereby certify that this correspondence is being electronically transmitted to the Patent and Trademark Office on the date indicated below in accordance with 37 CFR 1.8(a)(1)(i)(C).

July 6, 2010
Date of Transmission



Signature

Kethya Teuk
Typed or Printed Name of Person Signing Certificate

I. REAL PARTY IN INTEREST

SAP AG of Walldorf, Germany is the real party in interest as the owner of the above-identified application by virtue of an assignment from the inventors.

II. RELATED APPEALS AND INTERFERENCES

There are currently no other appeals or interferences, of which Appellant, Appellant's legal representative, or Assignee are aware, that will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Statement of the Status of All Claims. Per the Final Office Action mailed January 5, 2010 in this matter, claims 1-9, 18-20 and 26 stand rejected under 35 U.S.C. §102(b) as allegedly being anticipated by U.S. Patent No. 7,316,003 to Dulepet, et al. (hereinafter "Dulepet").

Appealed Claims. Appellants appeal the final rejections of independent claims 1, 6, 18 and 26. All of pending dependent claims 2-5, 7-9 and 19-20 depend directly or indirectly from one of claims 1, 6, 18 and 26 and therefore stand or fall with the outcome of the appeal of the independent claims.

IV. STATUS OF AMENDMENTS

All amendments in this matter have been entered. Per the Advisory Action of March 24, 2010, the Office considered the arguments regarding patentability presented in the Request for Reconsideration filed on March 16, 2010 in response to the Final Office Action of January 5, 2010.

V. SUMMARY OF CLAIMED SUBJECT MATTER

In accordance with 37 C.F.R. § 41.37(v), Appellants provide a brief summary of each independent claim involved in the appeal and each dependent claim argued separately, where each summary refers to the specification by page and line number and to the drawings by reference number. Appellants note that this "Summary of claimed subject matter" is provided only to assist the Board in identifying portions of the specification related to the particular claims. In the interest of brevity, each claim summary does not necessarily include all references to all relevant portions of the specifications and drawings. Accordingly, omission of any reference to the specification or to the drawings should not be construed in any way as an intent to relinquish claim scope or term, or as an implication or statement regarding the conformance with 35 U.S.C. §112. Appellants respectfully submit that the claims should not be construed as being limited to the embodiments described or referenced in any claim summary, and that other embodiments, as well as the Doctrine of Equivalents, may apply in determining claim scope. Similar terms in the claims as described below described similar structures and/or functions.

Independent claim 1. Claim 1 is directed to a computer-implemented method (¶[0060] – [0064]). The method includes providing a design-time translator and a run-time translator that both correspond to a defined page element (FIG. 3 step 300 and ¶[0040]. The run-time translator and design time translator are configured on a processor (¶[0061]).

During design-time for a page, the design-time translator is invoked for a page template (FIG. 3 and ¶¶[0040] – [0041]) including the defined page element having content components (FIG. 1 and ¶[0026]), The design-time invoking results in the defined page element in the page template being translated into a design-time representation of the content components in the page

(¶[0029]). The design-time representation is rendered in accordance with a predefined layout of a container for the content components (¶[0028]), and the page template is available to a plurality of remote users of a portal (see clients 200 in FIG. 2). The content components include a first content component and a second content component (¶¶[0011] – [0013]). The first content component is configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time (¶[0024]), such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time (¶[0046]).

The method further includes, during run-time for the page, invoking the run-time translator for the page template (FIG. 3 and ¶¶[0040] and [0042]), which results in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container (¶[0042]). The second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering (¶[0057]) in a content storage medium (¶¶[0057] – [0058]) to render the dynamic content of the second component rather than the tag used during design-time (¶[0042]).

Independent claim 6. Claim 6 is directed to an article comprising a machine-readable storage medium storing instructions operable to cause one or more machines to perform operations (¶[0061]). The operations include providing a design-time translator and a run-time translator that both correspond to a defined page element (FIG. 3 step 300 and ¶[0040]. The run-time translator and design time translator are configured on a processor (¶[0061]).

During design-time for a page, the design-time translator is invoked for a page template

(FIG. 3 and ¶¶[0040] – [0041]) including the defined page element having content components (FIG. 1 and ¶[0026]), The design-time invoking results in the defined page element in the page template being translated into a design-time representation of the content components in the page (¶[0029]). The design-time representation is rendered in accordance with a predefined layout of a container for the content components (¶[0028]), and the page template is available to a plurality of remote users of a portal (see clients 200 in FIG. 2). The content components include a first content component and a second content component (¶¶[0011] – [0013]). The first content component is configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time (¶[0024]), such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time (¶[0046]).

The operations further include, during run-time for the page, invoking the run-time translator for the page template (FIG. 3 and ¶¶[0040] and [0042]), which results in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container (¶[0042]). The second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering (¶[0057]) in a content storage medium (¶¶[0057] – [0058]) to render the dynamic content of the second component rather than the tag used during design-time (¶[0042]).

Independent claim 18. Claim 18 is directed to A portal system (¶[0028]) comprising a processor and memory configured to execute certain functions (¶[0059] and ¶[0061]). The functions include to provide a design-time translator and a run-time translator that both

correspond to a defined page element (FIG. 3 step 300 and ¶[0040]. The run-time translator and design time translator are configured on a processor (¶[0061]).

During design-time for a page, the design-time translator is invoked for a page template (FIG. 3 and ¶¶[0040] – [0041]) including the defined page element having content components (FIG. 1 and ¶[0026]), The design-time invoking results in the defined page element in the page template being translated into a design-time representation of the content components in the page (¶[0029]). The design-time representation is rendered in accordance with a predefined layout of a container for the content components (¶[0028]), and the page template is available to a plurality of remote users of a portal (see clients 200 in FIG. 2). The content components include a first content component and a second content component (¶¶[0011] – [0013]). The first content component is configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time (¶[0024]), such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time (¶[0046]).

The functions further include, during run-time for the page, to invoke the run-time translator for the page template (FIG. 3 and ¶¶[0040] and [0042]), which results in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container (¶[0042]). The second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering (¶[0057]) in a content storage medium (¶¶[0057] – [0058]) to render the dynamic content of the second component rather than the tag used during design-time (¶[0042]).

Independent claim 26. Claim 26 is directed to a computer-implemented method for selectively interpreting a portal page layout template (FIG. 3 ¶¶[0040] – [0042]. The method includes providing a design-time translator and a run-time translator (FIG. 3), the design-time translator and the run-time translator both corresponding to a same defined page element or placeholder (¶[0040]), and being invoked based on a current mode of operation (¶[0040]). The run-time translator and design time translator are configured on a processor (¶[0061]).

The method further includes translating a placeholder in a portal template during design-time into a representation of a container designed to present portal content using a single template file for both run-time and design-time (FIG. 3 and ¶[0041]). The container representation shows a layout context for the portal content that will be obtained and revealed at run-time (¶[0028]), and the container representation also directly presents dynamic content source information for the content container (¶[0027]).

The method further includes presenting a WYSIWYG portal layout editor using the container representation designed to present the portal dynamic content, where the WYSIWYG portal layout editor facilitates editing of the portal template and the resulting portal page (¶[0041]).

The method further includes obtaining portal dynamic content during run-time from a dynamic content source (¶[0042]), the placeholder in the portal template being translated by the runtime translator into a presentation of the container containing the obtained portal dynamic content component (FIG. 3 and ¶[0042]). The dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a dynamic content source (¶[0057]), providing a storage medium, to render the dynamic content rather than a tag used during design-time to represent the dynamic content

(¶[0042]).

The method further includes parsing and locating, by a run-time application, the placeholders in the template and replacing them with the run-time content components (¶[0042]), and at design-time, parsing and rendering the same template with a representation of the content components, in place of the actual run-time content components, to reveal the run-time layout during design of the template (¶[0041]).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Appellants request that the Board of Patent Appeals and Interferences review the following issues on appeal: The rejection under 35 U.S.C.102(b) of independent claims 1, 6, 18 and 26 are improper because Dulepet fails to disclose at least the following explicitly recited claim limitations, which are present in each of independent claims:

A) both a design time translator and a runtime translator that both correspond to a defined page element; and

B) the second component configured as dynamic content being determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time.

The patentability of claims 1-9, 18-20 and 26 should be considered together as the issues raised in this Appeal apply to all of the independent claims.

VII. ARGUMENT

Appellants respectfully submit that claims 1, 6, 18 and 26, and all claims that depend therefrom are allowable over the art currently of record in this matter. The issues presented for review are addressed below.

A. Dulepet fails to disclose each and every element recited in claims 1, 6, 18 and 26 and thus cannot anticipate these claims or any claims that depend therefrom under 35 U.S.C. §102.

Appellant respectfully submits that the Office's allegation that Dulepet anticipates the subject matter recited in claims 1, 6, 18 and 26 is not supported by the actual disclosure of Dulepet. In contrast to the Office's stated position, Dulepet neither discloses nor suggests several explicit limitations of claims 1, 6, 18 and 26, including but not necessarily limited to: 1) both a design time translator and a runtime translator that both correspond to a defined page element; 2) the second component configured as dynamic content being determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time.

To present a valid anticipation rejection under 35 U.S.C. §102, the Office must identify a single prior art reference in which "each and every element as set forth in the claim is found, either expressly or inherently described." MPEP §2131 quoting *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Furthermore, "the identical invention must be shown in as complete detail as is contained in the ... claim." MPEP §2131.01 quoting *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920

(Fed. Cir. 1989). The rejections of claims 1, 6, 18 and 26 over Dulepet fail to satisfy this burden with regards to the currently pending claims.

1. **Dulepet does not disclose a design time translator and a runtime translator that both correspond to a defined page element, as recited in independent claims 1, 6, 18 and 26 and the claims that depend therefrom.**

Claims 1, 6, 18 and 26 explicitly recite a design time translator and a runtime translator that both correspond to a defined page element. In particular, the claims recite that the run-time translator for the page template obtains the content components and translates the defined page element in the page template into a run-time presentation of the obtained one or more content components in accordance with a layout of the container.

Dulepet teaches a system and method for developing a web page that includes dynamically generated content. Specifically, Dulepet discloses an editor for developing a web page comprising dynamically generated content. See col. 5, lines 47-49 and the "editor" in FIGS. 2, 3 and 6. The editor interacts with an application database server (FIGS. 2 and 3) or otherwise known as a "design time engine" (see FIG. 6).

Dulepet teaches that an application server can execute JSP scripting elements within a page, "by generating dynamic page content (e.g. HTML) corresponding to the JSP scripting elements and replacing each JSP scripting element with the correspondingly dynamically generated content" (col. 1, lines 55-67). Dulepet also teaches that "a JSP enabled server may retrieve content from a database and/or enforce logic rules when fulfilling the corresponding JSP page request" (col. 2, lines 8-10. Appellants acknowledge that JSP scripting elements are eventually executed in a runtime environment, as the background of Dulepet suggests.

However, Dulepet fails to teach or suggest a runtime translator that corresponds to the same defined page element as a design-time translator. This feature of the claimed invention

provides a template-based page-layout capability such that the layout can execute both at runtime and design-time (see present specification at ¶[0025]). On the contrary, Dulepet teaches only an editor that develops a merged model representation of a dynamic web page (col. 12, lines 47-55), merging static and dynamic content in an HTML page that ostensibly can only be executed in a general runtime environment. Accordingly, Dulepet cannot be reasonably argued to disclose the limitation of both a design time translator and a runtime translator that both correspond to a defined page element.

2. **Dulepet does not disclose the second component configured as dynamic content being determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time, as recited in independent claims 1, 6, 18 and 26 and the claims that depend therefrom.**

Each of the independent claims 1, 6, 18 and 26 recite limitations of determining and obtaining dynamic content in parallel, at runtime using threading, with other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used at design-time. These claimed features allow a portal page during runtime to display dynamic content from a number of sources in parallel to improve performance. Dulepet fails to teach or suggest these limitations, nor does Dulepet disclose these limitations arranged as in the claims.

The Office asserts that Dulepet teaches obtaining multiple content items to multiple containers, and that concurrently running two or more tasks defines the term "threading." But, these alleged teachings Dulepet, while relevant to modern operating systems, do not suggest the limitations of using threading to determine and obtain dynamic content in parallel with other

dynamic content stored in blocks without ordering. In fact, Dulepet does not even mention the term "parallel" or any term similar. FIG. 5 in Dulepet clearly shows a recursive method for determining whether dynamic source code is present in the web page source code after the source code has been parsed. But FIG. 5, and the rest of Dulepet, is silent with respect to obtaining dynamic content in parallel at runtime using threading.

Accordingly, Dulepet fails to teach or suggest these limitations, and therefore fails to anticipate claims 1, 6, 18 and 26.

B. Dulepet cannot properly anticipate claims 2-5, 7-9, and 19-20 under 35 U.S.C. §102.

Claims 2-5, 7-9 and 19-20 are patentable over Dulepet based at least on their dependency from one of claims 1, 6, 18 and 26, the allowability of which is argued above.

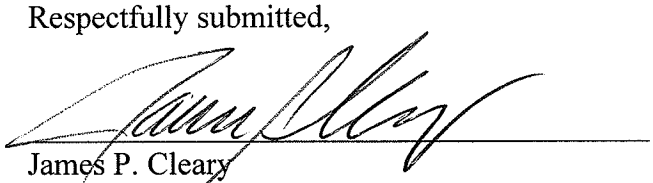
Concluding Comments

On the basis of the foregoing arguments, the pending claims are in condition for allowance. It is believed that all of the pending claims have been addressed in this paper. However, failure to address a specific rejection, issue or comment, does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above are not intended to be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper.

The Commissioner is hereby authorized to charge the fee for the filing of this Appeal Brief and any additional fees that may be due, or credit any overpayment of same, to Deposit Account No. 50-0311, Reference No. 34874-020/2003P00061US. If there are any questions regarding this reply, the Examiner is encouraged to contact the undersigned at the telephone number provided below.

Respectfully submitted,

Date: July 6, 2010



James P. Cleary
Reg. No. 45,843

Mintz, Levin, Cohn, Ferris, Glovsky and Popeo, P.C.
3580 Carmel Mountain Road, Suite 300
San Diego, CA 92130
Customer No. 64280
Tel.: 858/314-1533
Fax: 858/314-1501
jpcleary@mintz.com

VIII. CLAIMS APPENDIX: LISTING OF CLAIMS UNDER RULE 41.37(C)1(VIII)

1. (Previously Presented) A computer-implemented method comprising:

providing a design-time translator and a run-time translator that both correspond to a defined page element, the run-time translator and design time translator configured on a processor;

during design-time for a page, invoking the design-time translator for a page template including the defined page element having content components, said design-time invoking resulting in the defined page element in the page template being translated into a design-time representation of the content components in the page, the design-time representation being rendered in accordance with a predefined layout of a container for the content components, the page template being available to a plurality of remote users of a portal, the content components including a first content component and a second content component, the first content component configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time, such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time; and

during run-time for the page, invoking the run-time translator for the page template, said run-time invoking resulting in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container, wherein the second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without

ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time.

2. (Previously Presented) The method of claim 1, wherein said invoking the design-time translator further results in presentation of a WYSIWYG layout editor using the design-time representation of the one or more content components in the page.

3. (Previously Presented) The method of claim 2, wherein the said invoking the design-time translator further results in client-side scripting components being included in the design-time representation to form at least part of the WYSIWYG layout editor and enable adding a content component to a content container using a drag-and-drop action.

4. (Original) The method of claim 2, wherein the page template comprises a portal page template, and the WYSIWYG layout editor comprises a WYSIWYG portal page layout editor.

5. (Original) The method of claim 4, wherein the defined page element comprises a custom Java Server Page tag and the design-time translator and the run-time translator comprise Java Server Page tag handlers for the custom Java Server Page tag, and wherein the run-time translator obtains portal dynamic content according to the portal page template and the design-time translator does not.

6. (Previously Presented) An article comprising a machine-readable storage medium storing instructions operable to cause one or more machines to perform operations comprising: providing a design-time translator and a run-time translator that both correspond to a

defined page element, the run-time translator and design time translator configured on a processor;

during design-time for a page, invoking the design-time translator for a page template including the defined page element having-content components, said design-time invoking resulting in the defined page element in the page template being translated into a design-time representation of the content components in the page, the design-time representation being rendered in accordance with a predefined layout of a container for the content components, the page template being available to a plurality of remote users of a portal, the content components including a first content component and a second content component, the first content component configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time, such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time; and

during run-time for the page, invoking the run-time translator for the page template, said run-time invoking resulting in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container, wherein the second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time.

7. (Original) The article of claim 6, wherein translating the placeholder during design-time comprises adding code enabling editing of the portal page, the added code forming at least part of the WYSIWYG portal layout editor.

8. (Original) The article of claim 7, wherein the added code comprises client-side scripting that enables addition of a content component to a content container in the portal page using a drag-and-drop action.

9. (Original) The article of claim 6, wherein the placeholder comprises a custom Java Server Page tag, said translating the placeholder during design-time comprises invoking a design-time Java Server Page tag handler corresponding to the custom Java Server Page tag, and said translating the placeholder during run-time comprises invoking a run-time Java Server Page tag handler corresponding to the custom Java Server Page tag.

10-17 (Canceled).

18. (Previously Presented) A portal system comprising:

a processor; and

memory configured to:

provide a design-time translator and a run-time translator that both correspond to a defined page element, the run-time translator and design time translator configured on a processor;

during design-time for a page, invoke the design-time translator for a page template including the defined page element having content components, said design-time invoking

resulting in the defined page element in the page template being translated into a design-time representation of the content components in the page, the design-time representation being rendered in accordance with a predefined layout of a container for the content components, the page template being available to a plurality of remote users of a portal, the content components including a first content component and a second content component, the first content component configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time, such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time; and

during run-time for the page, invoke the run-time translator for the page template, said run-time invoking resulting in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container, wherein the second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time.

19. (Original) The system of claim 18, wherein the first tag handler interprets the portal page template by including client-side scripting that enables addition of a content component to a content container in the portal page template using a drag-and-drop action.

20. (Original) The system of claim 18, wherein the defined page element

comprises a custom Java Server Page tag and the design-time translator and the run-time translator comprise Java Server Page tag handlers for the custom Java Server Page tag, and wherein the run-time translator obtains portal dynamic content according to the portal page template and the design-time translator does not.

21-25. (Canceled).

26. (Previously Presented) A computer-implemented method for selectively interpreting a portal page layout template, the method comprising:

providing a design-time translator and a run-time translator, the design-time translator and the run-time translator both corresponding to a same defined page element or placeholder, and being invoked based on a current mode of operation, the run-time translator and design time translator configured on a processor;

translating a placeholder in a portal template during design-time into a representation of a container designed to present portal content using a single template file for both run-time and design-time, the container representation showing a layout context for the portal content that will be obtained and revealed at run-time, the container representation also directly presenting dynamic content source information for the content container;

presenting a WYSIWYG portal layout editor using the container representation designed to present the portal dynamic content, the WYSIWYG portal layout editor facilitating editing of the portal template and the resulting portal page;

obtaining portal dynamic content during run-time from a dynamic content source, the placeholder in the portal template being translated by the runtime translator into a presentation of the container containing the obtained portal dynamic content component, wherein

the dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a dynamic content source, providing a storage medium, to render the dynamic content rather than a tag used during design-time to represent the dynamic content; and

parsing and locating, by a run-time application, the placeholders in the template and replacing them with the run-time content components, and at design-time, parsing and rendering the same template with a representation of the content components, in place of the actual run-time content components, to reveal the run-time layout during design of the template.

IX. EVIDENCE APPENDIX

No additional evidence is filed with this Appeal Brief.

X. RELATED PROCEEDINGS APPENDIX

There are currently no other appeals or interferences, of which Appellant, Appellant's legal representative, or Assignee are aware, that will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.